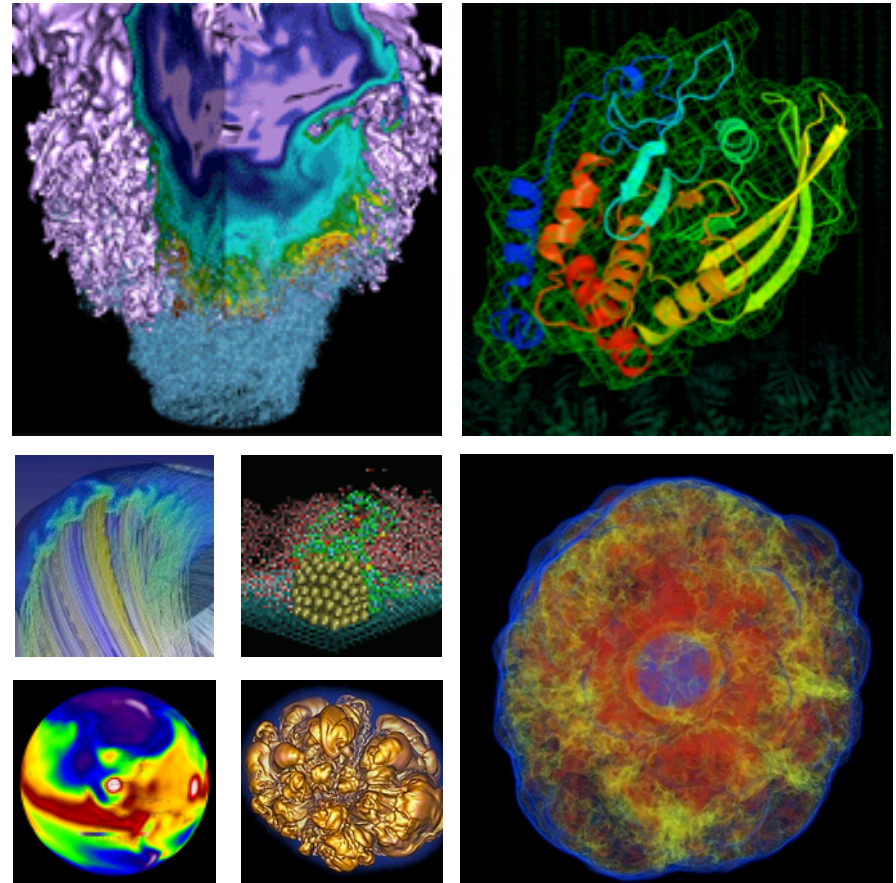


Using Craypat & Reveal on Cori



Rebecca Hartman-Baker
User Engagement Group Leader

Outline



- I. Profiling with Craypat**
- II. Using Reveal for OpenMP**



I. PROFILING WITH CRAYPAT

Profile of Cochise in the Chiricahua Mountains by Ken Bosma, <http://www.flickr.com/photos/kretyen/2879059366/>



U.S. DEPARTMENT OF
ENERGY

Office of
Science



I. Profiling with Craypat



- Introduction
- Simple profiling
- Full-service profiling

- **Craypat is Cray's Performance Analysis Tool**
- **Evaluate program behavior on Cray supercomputer**
 - Under any PrgEnv
- **Find hotspots, load imbalance, inefficiencies**
 - I/O, memory usage
 - MPI communications
 - Flops
 - Recommendation for rank reordering (sometimes)
- **Profiler with limited tracing abilities**
 - Tracing tools with better performance: MAP, VampirTrace

Simple Profiling with CrayPat



- `perftools-lite` module easier to use & does (almost) everything in `perftools`
- Compile code with `perftools-lite` module loaded
- Run code as normal
- **Output:**
 - Stdout & `*.rpt` file: report with execution time, memory high-water mark, aggregate FLOPS rate, top time-consuming user functions, MPI info, etc.
 - `*.ap2` file: can be viewed with Apprentice 2
 - (Possibly) `MPICH_RANK_REORDER` file

Example Output (Preamble)



```
CrayPat/X:  Version 6.4.0 Revision bc8f5bd  05/24/16 17:52:13
Experiment:                lite  lite/sample_profile
Number of PEs (MPI ranks):      64
Numbers of PEs per Node:       64
Numbers of Threads per PE:     1
Number of Cores per Socket:    68
Execution start time:  Thu Oct 13 09:30:31 2016
System name and speed:  nid04403  1401 MHz (approx)
Intel knl CPU Family:  6  Model: 87  Stepping: 1
MCDRAM: 7.2 GHz, 16 GiB available as quad, flat ( 0% cache)
```

```
Avg Process Time:    558.16 secs
High Memory:         1,899.7 MBytes      29.7 MBytes per PE
I/O Read Rate:       4.032070 MBytes/sec
I/O Write Rate:      3.618872 MBytes/sec
```

Example Output (Function Performance)



Table 1: Profile by Function Group and Function (top 10 functions shown)

Samp%	Samp	Imb.	Imb.	Group
		Samp	Samp%	Function
				PE=HIDE
100.0%	55,700.3	--	--	Total

49.3%	27,466.0	--	--	ETC

15.6%	8,679.3	1,276.7	13.0%	__cray_HCOSS_01
12.2%	6,821.2	1,167.8	14.8%	__cray_COS_V_01
8.9%	4,948.8	581.2	10.7%	_COS_Z
2.3%	1,285.3	335.7	21.0%	gotoblas_daxpy_k_knl
1.9%	1,071.1	235.9	18.3%	gotoblas_blas_memory_alloc_knl
1.9%	1,039.9	185.1	15.3%	gotoblas_dger_k_knl
=====				
35.5%	19,747.6	--	--	USER

23.0%	12,803.4	1,592.6	11.2%	intgrd_
7.6%	4,229.4	1,033.6	20.0%	dfshre_
3.1%	1,707.7	501.3	23.1%	drlhre_
=====				
14.7%	8,169.4	21,597.6	73.7%	MPI

14.7%	8,169.4	21,597.6	73.7%	MPI_Recv
=====				

Simple Profiling Recipe



- **Load/unload modules:**
 - `module unload darshan`
 - `module load perftools-base perftools-lite`
- **Compile and run your code as usual**

Full-Service Profiling



- **Motivation:**
 - Need more info than perftools-lite provides
 - Want to ignore certain subroutines
 - Focus on particular class of functions
 - Tracing rather than profiling
- **Super-deluxe profiling recipe**
- **pat_build options**

Super-Deluxe Profiling Recipe (1)



- **Load/unload modules:**
 - `module unload darshan`
 - `module load perftools-base perftools`
- **Compile code as usual, making sure to preserve object files**
- **`pat_build -O apa myapp`**
 - Generates executable called `myapp+pat`
- **Run `myapp+pat`**
 - Results in output file with name like `myapp+pat+#####.xf` or directory called `myapp+pat+#####`

Super-Deluxe Profiling Recipe (2)



- `pat_report myapp+pat+*.xf`
 - Generates `myapp+pat+*.apa`
- `pat_build -O myapp+pat+*.apa`
 - Generates executable called `myapp+apa`
- Run `myapp+apa`
- `pat_report myapp+apa+*.xf`

pat_build Options



- **pat_build -O apa myapp**
 - Craypat output for `myapp+pat` will be sampling to determine which subroutines can be ignored in full run. Additional file, `*.apa`, produced from `pat_report`
 - After this run, execute `pat_build -O *.apa` file to re-instrument `myapp+pat` into `myapp+apa` and run `myapp+apa` to get performance info
- **pat_build -g *tracegroup* myapp**
 - *tracegroup* is group of functions that can be automatically traced by CrayPat. Options include: `blas`, `fftw`, `mpi`, `netcdf`, `petsc`
- **pat_build -w myapp**
 - Do tracing experiment instead of profiling



II. PARALLELIZATION WITH CRAY REVEAL

“Happiness Revealed,” by Leonard Farshore, <https://flic.kr/p/9z7isd>



U.S. DEPARTMENT OF
ENERGY

Office of
Science



Cray Reveal



- Tool for porting to shared-memory or offload programming models
- Combine profiling info from Craypat and Cray compiler annotation to determine where to place OpenMP directives (generated automatically)
- **Works ONLY with Cray programming environment**

Using Cray Reveal



- 1. Compile code with Craypat instrumentation and create program library**
- 2. Run representative job**
- 3. Run Reveal**
- 4. Insert directives, consider loop reordering, and analyze performance from optimizations**

Cray Reveal Recipe (1)



- **Load/unload modules:**

- `module unload darshan`
- `module swap PrgEnv-intel PrgEnv-cray`
- `module load perftools-base perftools`

- **Compile & link with**

- `-h profile_generate` (to instrument), and
- `-h pl=/directory/path/myapp.pl` (for compiler feedback)

- **Instrument binary for tracing:**

- `pat_build -w ./myapp`
- Creates instrumented application: `myapp+pat`

Cray Reveal Recipe (2)



- **Run instrumented application (`myapp+pat`) as normal**
 - Ideally this is job requiring 5-15 mins runtime, performing important subroutines in similar proportions to typical run
- **This creates file called `myapp+pat+#####-##t.xf` (or directory `myapp+pat+#####-##t/` for large runs)**
- **Create report with loop statistics**
 - `pat_report myapp+pat+* > loops_report.txt`
 - Generates `.ap2` file & generates text report in output file

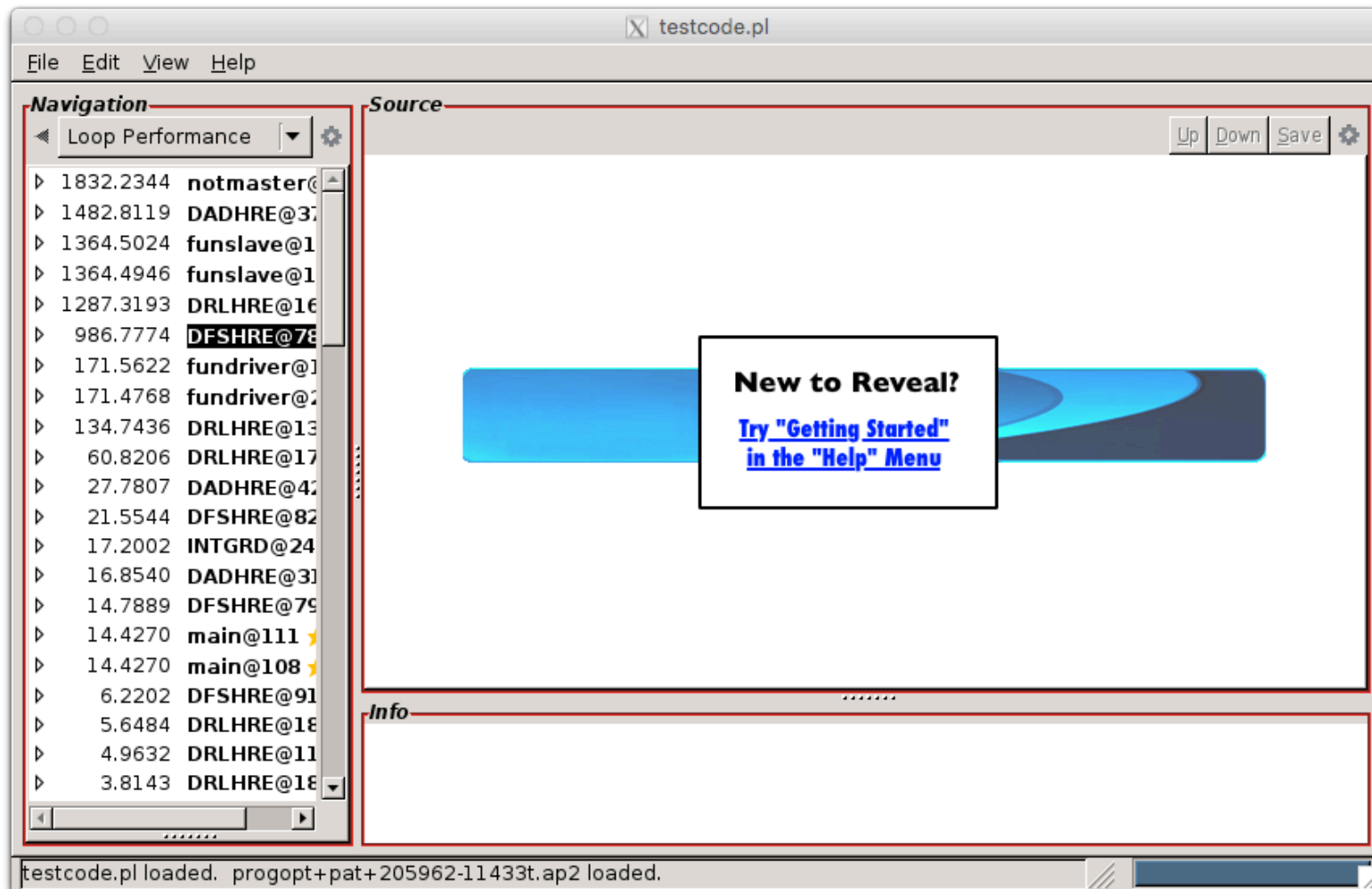
Cray Reveal Recipe (3)



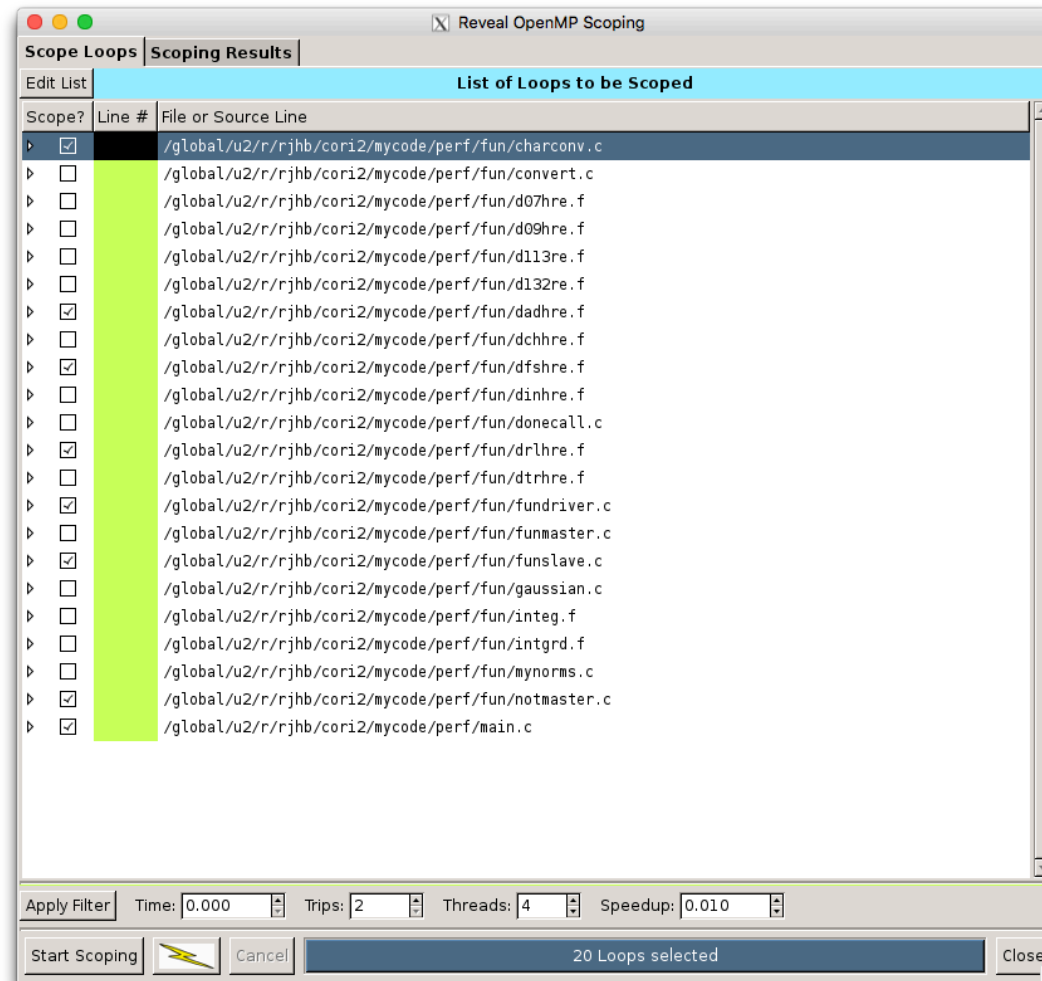
- **Run Reveal**

- `reveal /directory/path/myapp.pl` (compiler info only)
- `reveal /directory/path/myapp.pl myapp +#####t.ap2` (compiler + profiler info)

Opening Screen



Scoping Window



Compiler Annotations & Explanations



Source - /global/u2/r/rhbc/cori2/mycode/perf/fun/drlhre.f

```

129 20      CONTINUE
130 30      CONTINUE
131      DIFMAX = 0
132      RATIO = (G(1,3)/G(1,2))**2
133  DO 60 I = 1,NDIM
134      X(I) = CENTER(I) - HWIDTH(I)*G(1,2)
135      CALL FUNSUB(NDIM,X,NUMFUN,NULL(1,5))
136      X(I) = CENTER(I) + HWIDTH(I)*G(1,2)
137      CALL FUNSUB(NDIM,X,NUMFUN,NULL(1,6))
138      X(I) = CENTER(I) - HWIDTH(I)*G(1,3)
139      CALL FUNSUB(NDIM,X,NUMFUN,NULL(1,7))
140      X(I) = CENTER(I) + HWIDTH(I)*G(1,3)
141      CALL FUNSUB(NDIM,X,NUMFUN,NULL(1,8))
142      X(I) = CENTER(I)
143      DIFSUM = 0
144  DO 50 J = 1,NUMFUN
145      FRTHDF = 2* (1-RATIO)*RGNERR(J) - (NULL(J,7)+NULL(J,8)) +
146      +      RATIO* (NULL(J,5)+NULL(J,6))
147  C
148  C      Ignore differences below roundoff
149  C
150      IF (RGNERR(J)+FRTHDF/4.NE.RGNERR(J)) DIFSUM = DIFSUM +
151      +      ABS(FRTHDF)
152  DO 40 K = 1,4
153      NULL(J,K) = NULL(J,K) + W(K+1,2)*
154      +      (NULL(J,5)+NULL(J,6)) +
155      +      W(K+1,3)* (NULL(J,7)+NULL(J,8))
156  40      CONTINUE
157      BASVAL(J) = BASVAL(J) + W(1,2)* (NULL(J,5)+NULL(J,6)) +
158      +      W(1,3)* (NULL(J,7)+NULL(J,8))
159  50      CONTINUE
160      IF (DIFSUM.GT.DIFMAX) THEN
161          DIFMAX = DIFSUM
162          DIVAXN = I
163      END IF
164  60      CONTINUE
165      DIRECT = DIVAXN
166  C
167  C      Finish computing the rule values.
168  C
169  DO 90 I = 4,WLENG
170      CALL DFSHRE(NDIM,CENTER,HWIDTH,X,G(1,I),NUMFUN,FUNSUB,RGNERR,
171      +      NULL(1,5))

```

Info - Line 133

- A loop starting at line 133 was not vectorized because it contains a call to function "funsub" on line 135.
- A loop starting at line 144 was not vectorized because it contains a call to a subroutine or function on line 152.
- A loop starting at line 152 was partially vectorized with a single vector iteration.

Partial Success in Subroutine



```
File Edit View Help
Navigation
Program View
Loop@383
Loop@391
Loop@394
Loop@422
Loop@435
Loop@454
Loop@458
Loop@459
Loop@467
dchhre.f
dcuhre.f
dfshre.f
  DFLHRE
    Loop@69
    Loop@75
    Loop@79
    Loop@82
    Loop@91
    Loop@95
    Loop@111
dinhre.f
donecall.c
drlhre.f
  DRLHRE
    Loop@119
    Loop@125
    Loop@127
    Loop@133
    Loop@144
    Loop@152
    Loop@169
    Loop@172
    Loop@174
    Loop@182
    Loop@189
    Loop@191
dthhre.f
evalfun.c
fundriver.c
funmaster.c

Source - /global/u2/r/rjhb/cori2/mycode/perf/fun/dfshre.f
1 SUBROUTINE DFSHRE (NDIM, CENTER, HWIDTH, X, G, NUMFUN, FUNSUB, FULSMS,
2 + FUNVLS)
3 C***BEGIN PROLOGUE DFSHRE
4 C***KEYWORDS fully symmetric sum
5 C***PURPOSE To compute fully symmetric basic rule sums
6 C***AUTHOR Alan Genz, Computer Science Department, Washington
7 C State University, Pullman, WA 99163-1210 USA
8 C***LAST MODIFICATION 88-04-08
9 C***DESCRIPTION DFSHRE computes a fully symmetric sum for a vector
10 C of integrand values over a hyper-rectangular region.
11 C The sum is fully symmetric with respect to the center of
12 C the region and is taken over all sign changes and
13 C permutations of the generators for the sum.
14 C
15 C ON ENTRY
16 C
17 C NDIM Integer.
18 C Number of variables.
19 C CENTER Real array of dimension NDIM.
20 C The coordinates for the center of the region.
21 C HWIDTH Real Array of dimension NDIM.
22 C HWIDTH(I) is half of the width of dimension I of the region.
23 C X Real Array of dimension NDIM.
24 C A work array.
25 C G Real Array of dimension NDIM.
26 C The generators for the fully symmetric sum. These MUST BE
27 C non-negative and non-increasing.
28 C NUMFUN Integer.
29 C Number of components for the vector integrand.
30 C FUNSUB Externally declared subroutine.
31 C For computing the components of the integrand at a point X.
32 C It must have parameters (NDIM, X, NUMFUN, FUNVLS).
33 C Input Parameters:
34 C X Real array of dimension NDIM.
35 C Defines the evaluation point.
36 C NDIM Integer.
```

Successful Scoping



Navigation

- Program View
- Loop@383
- Loop@391
- Loop@394
- Loop@422
- Loop@435
- Loop@454
- Loop@458
- Loop@459
- Loop@467
- dfshre.f
- dcuhre.f
- dfshre.f
- DFSHRE
- Loop@69
- Loop@75
- Loop@79
- Loop@82
- Loop@91
- Loop@95
- Loop@111
- dinhre.f
- donecall.c
- drhre.f
- DRLHRE
- Loop@119
- Loop@125
- Loop@127
- Loop@133
- Loop@144
- Loop@152
- Loop@169
- Loop@172
- Loop@174
- Loop@182
- Loop@189
- Loop@191
- dtrhre.f
- evalfun.c
- fundriver.c
- funmaster.c

Source - /global/u2/r/rjhb/cori2/mycode/perffun/dfshre.f

```
71 C CONTINUE
72 C
73 C Compute centrally symmetric sum for permutation of G
74 C
SVpr2 75 20 DO 30 I = 1,NDIM
76 X(I) = CENTER(I) + G(I)*HWIDTH(I)
77 CONTINUE
L 78 40 CALL FUNSUB(NDIM,X,NUMFUN,FUNVLS)
SVr2 79 DO 50 J = 1,NUMFUN
80 FULSMS(J) = FULSMS(J) + FUNVLS(J)
81 CONTINUE
S 82 DO 60 I = 1,NDIM
83 G(I) = - G(I)
84 X(I) = CENTER(I) + G(I)*HWIDTH(I)
85 IF (G(I).LT.0) GO TO 40
86 60 CONTINUE
87 C
88 C Find next distinct permutation of G and loop back for next sum.
```

Reveal OpenMP Scoping

Scope Loops Scoping Results

dfshre.f: Loop@75

Name	Type	Scope	Info
i	Scalar	Private	
center	Array	Shared	

First/Last Private

☐ Enable FirstPrivate

☐ Enable LastPrivate

Reduction

None

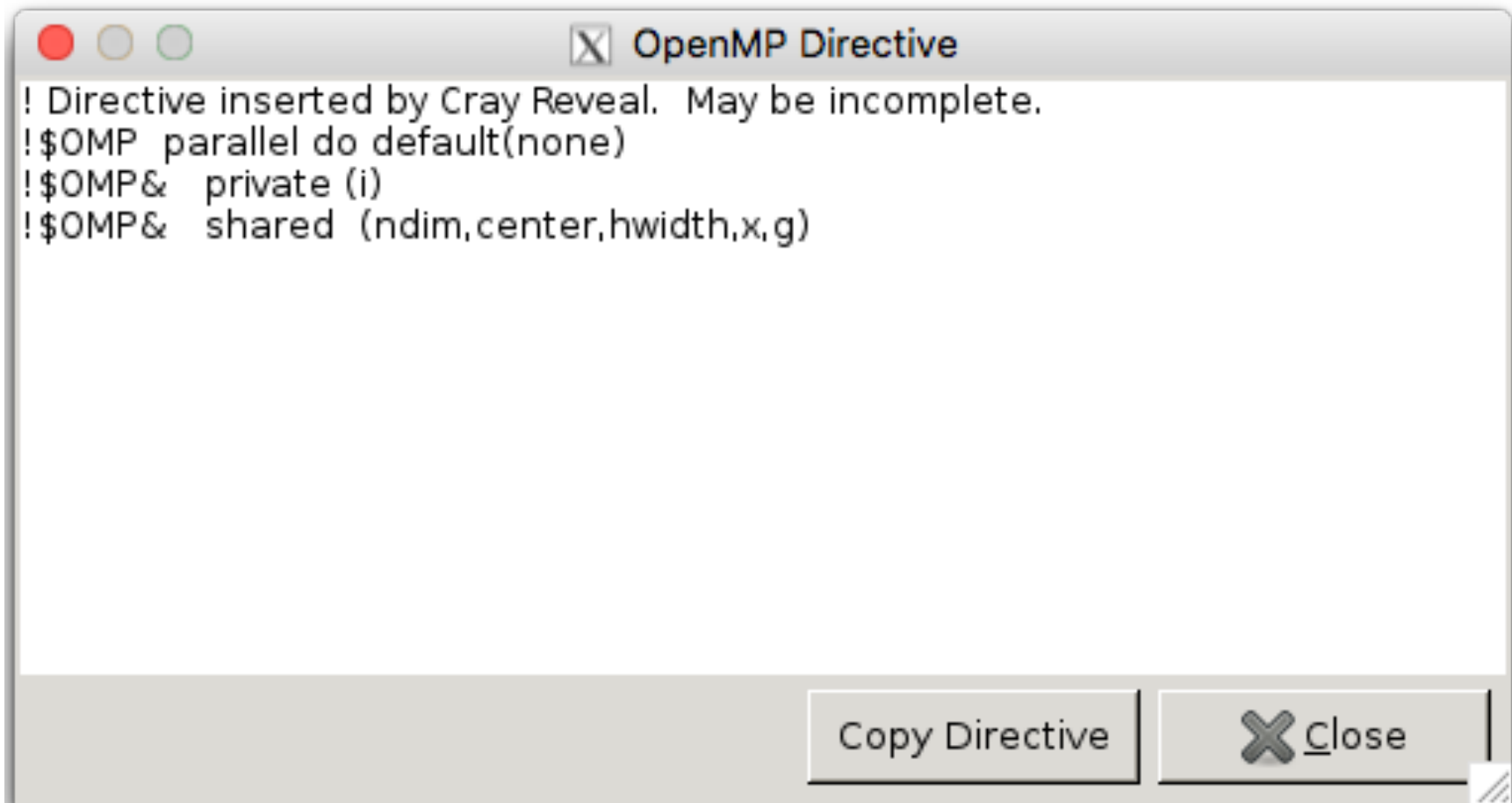
Find Name:

Insert Directive Show Directive Close

Info - Line 75

- A loop starting at line 75 was scoped
- A loop starting at line 75 was partially
- A loop starting at line 75 was unrolled

Directives Generated by Reveal



Unsuccessful Scoping



Reveal OpenMP Scoping

Scope Loops | Scoping Results | dadhre.f: Loop@422

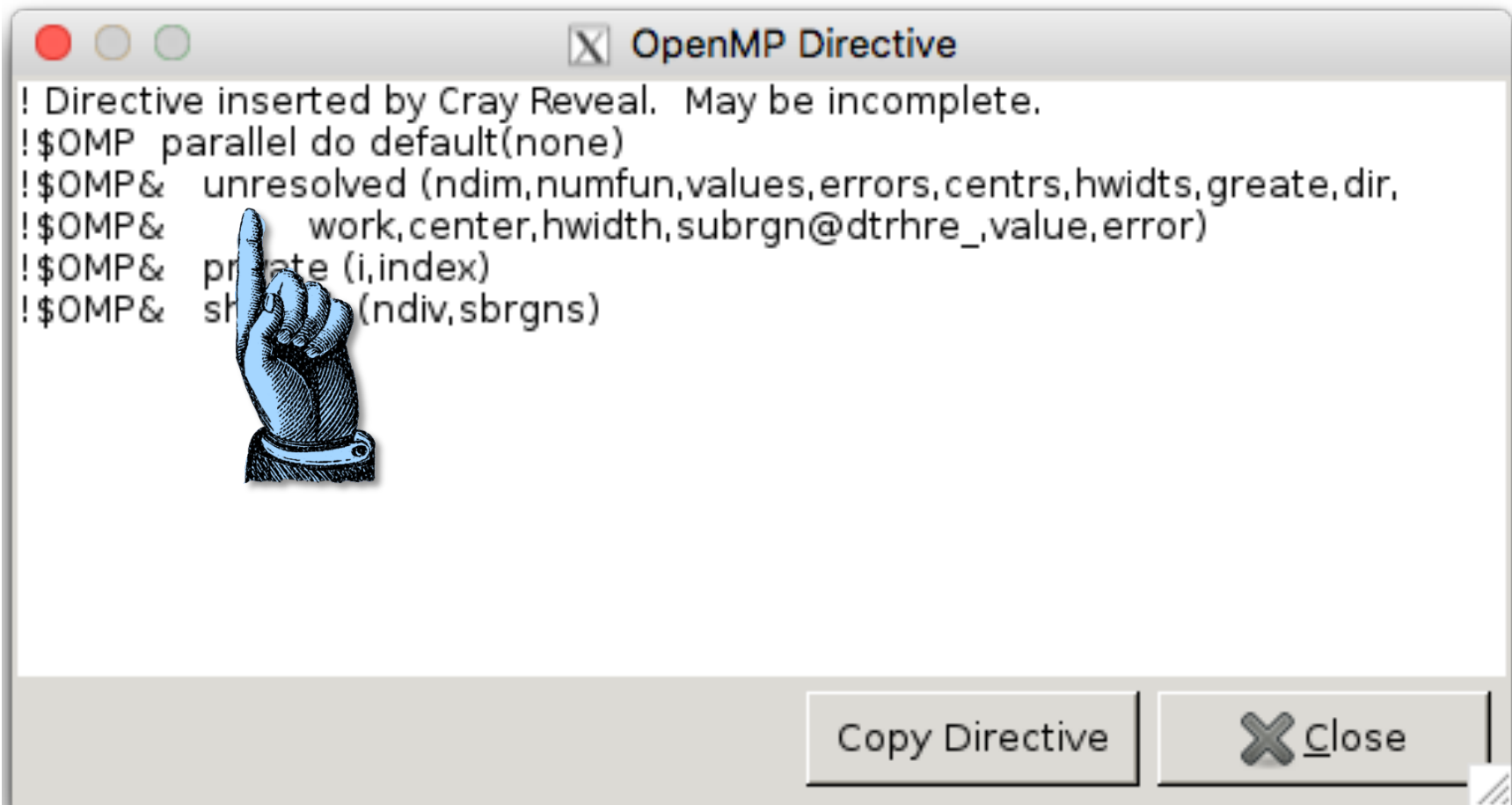
Name	Type	Scope	Info
centrs I	Array	Unresolved	FAIL: Possible recurrence involving this object.
dir I	Array	Unresolved	WARN: Assuming no conflict in scatter. FAIL: Possible recurrence involving this object.
error I	Scalar	Unresolved	FAIL: Possible recurrence involving this object.
errors I	Array	Unresolved	FAIL: Possible recurrence involving this object.
greate I	Array	Unresolved	WARN: Assuming no conflict in scatter. FAIL: Possible recurrence involving this object.
hwidths I	Array	Unresolved	FAIL: Possible recurrence involving this object.
ndim I	Scalar	Unresolved	FAIL: conflicting requirements, unable to scope.
numfun I	Scalar	Unresolved	FAIL: conflicting requirements, unable to scope.
subrgn@drhre_ I	Scalar	Unresolved	FAIL: Possible recurrence involving this object.
value I	Scalar	Unresolved	FAIL: Possible recurrence involving this object.
values I	Array	Unresolved	FAIL: Possible recurrence involving this object.
work I	Array	Unresolved	WARN: Assuming no conflict in scatter. FAIL: Possible recurrence involving this object.
center I	Array	Private	WARN: LastPrivate of array may be very expensive.
hwidth I	Array	Private	WARN: LastPrivate of array may be very expensive.
i	Scalar	Private	
index	Scalar	Private	
ndiv	Scalar	Shared	
sbrgns	Scalar	Shared	

First/Last Private: ☐ Enable FirstPrivate ☐ Enable LastPrivate

Reduction:

Find Name:

Unsuccessful Scoping Directive



Cray Reveal Recipe (4)



- **Insert directives**
- **Examine compiler feedback to determine potential loop reordering**
 - E.g., row- vs. column-ordered memory access patterns
 - Moving conditionals outside of loops
 - Cray compiler good at loop optimizations but requires some human help at times
- **Analyze performance after optimizations**
 - (Lather, rinse, repeat)



National Energy Research Scientific Computing Center